We claim:

1. In a computer system, a method of expressing a code invocation contract in computer program code, wherein the computer program code is stored on a computer-readable medium, and wherein the computer program code is operable to cause a computer to perform according to instructions in the computer program code, the method comprising:

adding one or more keywords to a function interface for a function in the computer program code, wherein the one or more keywords define the code invocation contract for the function, the contract having one or more contract requirements;

wherein the one or more contract requirements are independent of function call context.

2. The method of claim 1 wherein the one or more contract requirements include a postcondition.

3. The method of claim 1 wherein the one or more contract requirements include a precondition.

4. The method of claim 1 wherein the one or more contract requirements include one or more conditions to be fulfilled by one or more calling functions.

5. The method of claim 1 wherein the one or more contract requirements include one or more conditions to be fulfilled by one or more called functions.

6. A computer-readable, program-carrying medium having carried thereon computer program code having keywords added to the computer program code according to the method of claim 1.

7. In a computer system, a method of annotating computer program code stored on a computer-readable medium, wherein the computer program code is operable to cause a computer to perform according to instructions in the computer program code, the method comprising:

inserting one or more code annotations only at one or more annotation targets;

wherein each of the one or more annotation targets is of an annotation target category selected from a group consisting of: global variable, formal parameter of a function, return value of a function, user-defined data type.

8. The method of claim 7 wherein the inserting one or more code annotations only at one or more annotation targets comprises inserting default annotations.

9. The method of claim 7 wherein the one or more code annotations include a property.

10. The method of claim 9 wherein the property includes an implicit dereference qualifier.

11. The method of claim 9 wherein the property is read only.

12. The method of claim 9 wherein the property is a return value property.

13. The method of claim 9 wherein the property indicates a characteristic of a buffer.

14. The method of claim 13 wherein the characteristic is a readable extent of the buffer.

15. The method of claim 13 wherein the characteristic is a writable extent of the buffer.

16. The method of claim 9 wherein the property indicates a location for a buffer pointer.

17. The method of claim 7 wherein the one or more code annotations include a qualifier.

18. The method of claim 17 wherein the qualifier is a precondition qualifier.

19. The method of claim 17 wherein the qualifier is a postcondition qualifier.

20. The method of claim 17 wherein the qualifier is an except qualifier.

21. The method of claim 7 wherein the one or more code annotations include an annotation prefix.

22. The method of claim 21 wherein the annotation prefix comprises a dereference prefix at a reference parameter, and wherein the dereference prefix is used to specify properties of an object referenced by the reference parameter.

23. The method of claim 7 wherein the inserting one or more code annotations only at one or more annotation targets comprises inserting a first code annotation at a reference parameter, and wherein the first code annotation applies to the reference parameter itself.

24. In a computer system, a method of annotating computer program code stored on a computer-readable medium, wherein the computer program code is operable to cause a computer to perform according to instructions in the computer program code, the method comprising:

inserting an annotation at a value having a value type in the computer program code;

wherein the annotation is a keyword indicating that the value has usability properties sufficient to allow a function to rely on the value, wherein the usability properties depend on the value type.

25. The method of claim 24 wherein the value is a formal parameter of the function.

26. The method of claim 24 wherein the value is a return value.

27. The method of claim 24 wherein the value type is selected from a group comprising: scalar, void, pointer, user-defined type, struct.

28. The method of claim 24 wherein the value is a reference parameter.

29. The method of claim 24 wherein the value is a pointer, wherein an object pointed to by the pointer has one or more readable elements, the one or more readable elements of the object each having usability properties sufficient to allow the function to rely on the one or more readable elements.

30. In a computer system, a method of annotating computer program code stored on a computer-readable medium, wherein the computer program code is operable

to cause a computer to perform according to instructions in the computer program code, the method comprising:

inserting an annotation having an argument in the computer program code, wherein the annotation annotates a value having a first value type, and wherein usability properties of the value are dependent on the first value type;

wherein the annotation indicates that the value has usability properties that depend on the properties of a second value type denoted by the argument of the annotation.

31. The method of claim 30 wherein the first value type is a legacy value type.

32. The method of claim 30 wherein the first value type is void *.

33. The method of claim 30 wherein the first value type is char *.

34. In a computer system, a method of annotating computer-executable program code stored on a computer-readable medium, the method comprising:

adding an annotation to the computer program code, wherein the annotation describes a characteristic of a buffer; and

including a size parameter with the annotation, wherein the size parameter describes a portion of the buffer to which the characteristic applies, and wherein the size parameter is operable to describe the portion of the buffer using a size specification selected from a group of plural different size specifications.

35. The method of claim 34 wherein the group of plural different size specifications comprises: byte count, element count, end pointer location, internal pointer location, sentinel position.

36. The method of claim 34 wherein the annotation indicates the extent to which the buffer is readable.

37. The method of claim 34 wherein the annotation indicates the extent to which the buffer is writable.

38. In a computer system, a method of annotating computer-executable program code stored on a computer-readable medium, the method comprising:

adding an annotation to the computer program code, wherein the annotation comprises an arrangement of lexical components, and wherein the arrangement consists of:

an optional precondition qualifier or postcondition qualifier; followed immediately thereafter by

an optional exception qualifier; followed immediately thereafter by

one or more optional dereference qualifiers; followed immediately thereafter by

a property.

39. A computer programmed as a source code annotation system, the computer comprising:

a memory storing program code for the source code annotation system; and

a processor for executing the program code for the source code annotation system;

wherein the program code for the source code annotation system comprises:

program code for instructing a computer to add one or more annotations to one or more annotation targets in high-level language source code, wherein the one or more annotations each comprise an arrangement of one or more lexical components, the arrangement consisting of an optional precondition

qualifier or postcondition qualifier, followed immediately thereafter by an optional exception qualifier, followed immediately thereafter by one or more optional dereference qualifiers, followed immediately thereafter by a property.

40. The computer of claim 39 wherein the high-level language is an object-oriented programming language.